DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm

Luiz F. Bittencourt\textsuperscript{1}, Rizos Sakellariou\textsuperscript{2}, and Edmundo R. M. Madeira\textsuperscript{1}

bit@ic.unicamp.br
rizos@cs.man.ac.uk
edmundo@ic.unicamp.br

\textsuperscript{1}Institute of Computing - University of Campinas (UNICAMP) - Brazil
\textsuperscript{2}School of Computer Science - The University of Manchester - United Kingdom
Summary

1. Introduction
   - Task Scheduling
   - HEFT heuristic

2. Proposed algorithms
   - HEFT with lookahead
   - HEFT with priority list change

3. Simulation Results

4. Conclusion

5. Future works
Introduction

- Grids are heterogeneous, non-dedicated and distributed computing environments.
- Directed Acyclic Graphs (DAGs) often used to capture precedence constraints of a number of applications, such as workflows.
- Grids usually have high processing power and storage capacities, matching with e-Science application requirements.
- Consequently, an increasing interest in DAG scheduling heuristics in heterogeneous systems.
Dependent Task Scheduling

- Dependent task scheduling: how to distribute dependent tasks among the available resources?
- NP-Complete problem.
- A workflow is composed of dependent tasks, where each task depends on data computed by their predecessors.
- A workflow is represented by a Directed Acyclic Graph (DAG).
- Task scheduling can have different objective functions:
  - Minimize processes execution time
  - Maximize throughput
  - Minimize network traffic
  - Etc.
Dependent Task Scheduling

- Dependent task scheduling: how to distribute dependent tasks among the available resources?
- NP-Complete problem.
- A workflow is composed of dependent tasks, where each task depends on data computed by their predecessors.
- A workflow is represented by a Directed Acyclic Graph (DAG).
- Task scheduling can have different objective functions:
  - Minimize processes execution time \(\Leftarrow\) We focus on this one.
  - Maximize throughput
  - Minimize network traffic
  - Etc.
DAG Example

![DAG Diagram](image_url)

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>19</td>
<td>28</td>
<td>36</td>
<td>15</td>
<td>30</td>
<td>33</td>
<td>12</td>
<td>13</td>
<td>41</td>
</tr>
<tr>
<td>R2</td>
<td>41</td>
<td>46</td>
<td>34</td>
<td>25</td>
<td>50</td>
<td>35</td>
<td>20</td>
<td>22</td>
<td>68</td>
</tr>
<tr>
<td>R3</td>
<td>34</td>
<td>20</td>
<td>62</td>
<td>37</td>
<td>54</td>
<td>59</td>
<td>21</td>
<td>24</td>
<td>73</td>
</tr>
</tbody>
</table>
```
HEFT Heuristic

- List scheduling heuristic: priority + local optimal decisions.
- Most often cited and used: simplicity and good results in general.
- Locally optimal decision for a task may not necessarily be good for its successors in the DAG.

→ Explore a larger search space to check how such decisions would affect the tasks descendants.
HEFT Heuristic

- Uses average computation and communication costs to rank tasks.
- The priority is given by the \( \text{rank}_u \) attribute:
  \[
  \text{rank}_u(t_i) = \overline{w}_i + \max_{t_j \in \text{succ}(t_i)} (c_{i,j} + \text{rank}_u(t_j)),
  \]
- Tasks are scheduled in descending order of their \( \text{rank}_u \) value on the resource which gives the smallest estimated finish time (EFT) for each task.
Use lookahead information to foresee how decisions affect other tasks.

- Try each task on every resource and check the impact on its children.

- Increase in complexity.

  - HEFT is simple and fast: increase in execution time is affordable.
Scheduling using HEFT with lookahead

Proposed algorithms

HEFT with lookahead

Example

Bittencourt, Sakellariou, and Madeira

Scheduling using HEFT with lookahead

PDP‘10, Feb. 17th-19th
Scheduling using HEFT with lookahead

Proposed algorithms

HEFT with lookahead

Example

Bittencourt, Sakellariou, and Madeira

PDP’10, Feb. 17th-19th
Example

Scheduling using HEFT with lookahead

Proposed algorithms

HEFT with lookahead

Bittencourt, Sakellariou, and Madeira

PDP’10, Feb. 17th-19th 11 / 30
Example
Scheduling using HEFT with lookahead

Proposed algorithms

HEFT with lookahead

Example

![Diagram showing task dependencies and resource scheduling]

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19</td>
<td>28</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>46</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>20</td>
<td>62</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>37</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>33</td>
<td>59</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>31</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>41</td>
<td>17</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>73</td>
</tr>
</tbody>
</table>

Bittencourt, Sakellariou, and Madeira (2010)
Example
When trying a task $t$ on each resource, some children may not become ready to be scheduled.

- Consider only already scheduled parents when computing children’s EFT.
HEFT with lookahead - issues

- Several ways of selecting a resource for $t$ based on its children EFTs. We evaluate two ways:
  - Minimum maximum EFT among all $t$’s children.
  - Weighted average EFTs of $t$’s children.

$$EFT_W = \frac{\sum_{t_j \in L} (rank_u(t_j) \times EFT_{t_j})}{\sum_{t_j \in L} rank_u(t_j)},$$

where $L$ is the lookahead set containing the tasks to have their EFT considered in the resource selection.
1. rank tasks using the \( rank_u \)
2. while there are unscheduled tasks do
3. \( t \leftarrow \) unscheduled task with highest \( rank_u \)
4. \( L \leftarrow \) children of \( t \)   
   // \( L \) is the lookahead set
5. for all resource \( r_i \) in the resources set \( R \) do
6. schedule \( t \) on \( r_i \)
7. schedule all tasks of \( L \) using HEFT
8. \( EFT_{r_i} \leftarrow \) maximum EFT for tasks in \( L \)
9. return to the schedule state at the beginning of this loop
10. schedule \( t \) on \( r_i \) such that \( EFT_{r_i} \leq EFT_{r_k} \forall r_k \in R \)

- The version that makes use of the weighted average requires a modification in line 8 to assign \( EFT_W \) to \( EFT_{r_i} \).
Another approach: relax priorities given by HEFT.

Invert scheduling order given by $\text{rank}_u$ trying to achieve a better schedule.

Sort of “priority list lookahead”.

Use the lookahead algorithm twice:
  - considering tasks in the $\text{rank}_u$ order.
  - inverting the order of the first 2 tasks.
Priority list change overview

Let $t$ be the unscheduled ready task with highest priority and $t_1$ the second unscheduled ready task with the highest priority.

1. $L \leftarrow t_1 \cup suc(t) \cup suc(t_1)$
2. Execute the lookahead algorithm using $L$ as the lookahead set.
3. Exchange $t$ and $t_1$ and repeat these steps.
4. The order which gives the best overall result is then maintained.
Simulation

- Average makespan for the lookahead and priority list change, both with the minimize maximum EFT and weighted average variations.
- Execution time of each algorithm.
- 2 and 10 heterogeneous resources in the target environment.
- The resources were fully connected by a heterogeneous network.
- CCR (communication to computation ratio) 0.5, 1.0, and 2.0
- 5 DAGs representing real-world applications.
DAGs

(a) Montage

(b) LIGO

(c) AIRSN

(d) Chimera-1

(e) Chimera-2
Average Makespan - Montage

- Highest makespan improvements over HEFT were obtained by the lookahead algorithm using the minimize maximum EFT approach.
- Improvements from 1.55% (2 resources and $CCR = 0.5$) to 15.2% (10 resources and $CCR = 2.0$).
## Simulation Results

### Average Makespan - AIRSN

#### AIRSN - 2 resources

<table>
<thead>
<tr>
<th>CCR</th>
<th>HEFT</th>
<th>Lookahead</th>
<th>Lookahead-Wavg</th>
<th>PriorityChange</th>
<th>PriorityChange-Wavg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1200</td>
<td>1250</td>
<td>1300</td>
<td>1350</td>
<td>1400</td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>1.0</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### AIRSN - 10 resources

<table>
<thead>
<tr>
<th>CCR</th>
<th>HEFT</th>
<th>Lookahead</th>
<th>Lookahead-Wavg</th>
<th>PriorityChange</th>
<th>PriorityChange-Wavg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>400</td>
<td>450</td>
<td>500</td>
<td>550</td>
<td>600</td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>1.0</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The priority list change algorithm using the weighted average approach performed better with the AIRSN DAG.
- With 10 resources, the results are similar for both algorithms when using the weighted average approach, with improvements around 6%. 
Similar makespans when having 2 resources.

The best result was given by the lookahead with weighted average when having 10 resources, resulting in an average makespan 7.2% lower than HEFT’s makespan.
Similar results for $CCR = 0.5$ and $CCR = 1.0$.

When the CCR is higher, the makespan improvement of the lookahead algorithm with weighted average is around 20%.
Average Makespan - Chimera-2

- Similar to Chimera-1 results.
- Best improvement: 14.4% with 10 resources and $CCR = 2.0$. 
Lookahead versions took more time, as expected.

None of the algorithms has a prohibitively high execution time when compared to HEFT.

Lookahead alone (no priority list) versions: at most four times more than HEFT for Chimera-2, which has a high edge density.
Conclusion

- We presented 4 approaches to schedule DAGs using lookahead information and the HEFT algorithm.
- Simulations with DAGs corresponding to real-world applications show the proposed lookahead can significantly improve HEFT’s results.
- Better results with higher communication.
Future works

- Different approaches to look ahead.
- Further work on the feasibility of a higher lookahead depth.
- Effect of the lookahead strategies to other similar DAG scheduling heuristics.
Thanks

Thank you. Questions?

Acknowledgements: